Each SMART File is listed in one or more of the categories below. Following the categories is a list of each model with a brief description of its application and the key modules or constructs used.

Animation
Smarts06, Smarts17, Smarts 33, Smarts39, Smarts41, Smarts64, Smarts116

Arrivals
Smarts34, Smarts49, Smarts50, Smarts51, Smarts53, Smarts71, Smarts81, Smarts99, Smarts108, Smarts111

Basic Concepts
Smarts04, Smarts16, Smarts33, Smarts42

Batching
Smarts01, Smarts23, Smarts25, Smarts43

Blocking
Smarts21, Smarts24, Smarts50, Smarts57, Smarts60, Smarts76, Smarts85, Smarts87, Smarts110

Comparative Analysis
Smarts08, Smarts52

Continuous Processes
Smarts14, Smarts93, Smarts94, Smarts95

Control Logic
Smarts21, Smarts24, Smarts40, Smarts50, Smarts51, Smarts70, Smarts72, Smarts76, Smarts87, Smarts88, Smarts91, Smarts107

Conveyors
Smarts09, Smarts10, Smarts11, Smarts12, Smarts13, Smarts57, Smarts61, Smarts62, Smarts67, Smarts71, Smarts72, Smarts73, Smarts87, Smarts103

Decision Logic
Smarts05, Smarts07, Smarts13, Smarts22, Smarts27, Smarts28, Smarts62, Smarts67, Smarts69, Smarts71, Smarts74, Smarts91

Examples Using New Arena Modules
Smarts86, Smarts87, Smarts88, Smarts89, Smarts90, Smarts91, Smarts92, Smarts93, Smarts94, Smarts95

External Data Files
Smarts30, Smarts58, Smarts79, Smarts84

Job Prioritization
Smarts29, Smarts31, Smarts36, Smarts77, Smarts101

OLE
Smarts115, Smarts116

On-Screen Display of System Status
Smarts02, Smarts03, Smarts14, Smarts15, Smarts21, Smarts56, Smarts61, Smarts63, Smarts69, Smarts92

Queues
Smarts27, Smarts31, Smarts34, Smarts36, Smarts40, Smarts60, Smarts77, Smarts82

Resources
Smarts18, Smarts20, Smarts24, Smarts28, Smarts29, Smarts32, Smarts35, Smarts38, Smarts41, Smarts51, Smarts54, Smarts55, Smarts56, Smarts57, Smarts60, Smarts63, Smarts65, Smarts69, Smarts74, Smarts75, Smarts78, Smarts85, Smarts86, Smarts101, Smarts105, Smarts113

Run Length
Smarts96, Smarts97, Smarts100, Smarts108, Smarts112, Smarts114

Simultaneous Processing
Smarts23, Smarts83

Sequences
Smarts37, Smarts80, Smarts102

Statistics
Smarts58, Smarts75, Smarts81, Smarts105, Smarts106, Smarts109

Transporters
Smarts19, Smarts26, Smarts44, Smarts45, Smarts46, Smarts47, Smarts48, Smarts59, Smarts66, Smarts87, Smarts104

User Interaction
Smarts51, Smarts89, Smarts90, Smarts91, Smarts98, Smarts112

Variables & Expressions
Smarts15, Smarts49, Smarts54, Smarts68, Smarts70, Smarts75, Smarts78, Smarts98

Visual Basic
Smarts117, Smarts118, Smarts119, Smarts120, Smarts121, Smarts122, Smarts123, Smarts124, Smarts125, Smarts126, Smarts127

The SMART Files Library

SMARTS01
This model shows two types of entities entering a system, then batching into groups of three according to their type. The batches then leave the system and are counted. The purpose of the model is to demonstrate the use of the **Batch** module and a **Counter Set**.
Modules used: Arrive, Enter, Batch, Leave, Depart, Simulate, Sets, Variable animation.

SMARTS02
This model demonstrates how to animate the number of entities in a queue in an on-screen plot. The focus of the model is the **Plot** animation.
Modules used: Arrive, Server, Depart, Simulate, Plot animation.

SMARTS03
This model shows how to animate an analog and digital clock, and the date. The model utilizes the **Clock** animation and the **Date** animation.
Modules used: Arrive, Depart, Simulate, Clock animation, Date animation.

SMARTS04
This example demonstrates how to set up a basic model. It uses the **Arrive** module, the **Server** module, a **Depart** module, and a **Simulate** module.
Modules used: Arrive, Server, Depart, Simulate.

SMARTS05
In this model an entity selects which resource to go to based on a probability. The focus of the example is the proper use of the **Chance** module.
Modules used: Arrive, Enter, Chance, Process, Leave, Depart, Simulate.

SMARTS06

This model shows how to change an entities picture after it has completed processing. The example demonstrates how to use the *Animate* section of the **Server** module, and explains how to use the *Entity Picture* section of the **Simulate** module.

Modules used: Arrive, Server, Depart, Simulate.

SMARTS07

In this model an entity selects which resource to go to based on which one's queue is shorter. The focus of the example is the proper use of the **Choose** module.

Modules used: Arrive, Enter, Choose, Process, Leave, Depart, Simulate.

SMARTS08

When an entity is created in this model, it is assigned an estimate for the time it will spend in the system. This estimate is then compared to the actual value, and a plot is generated showing the difference. The model uses the *Time in Queue* section of the **Server** module, and the **Assign** module.

Modules used: Arrive, Server, Assign, Dispose, Simulate, Plot animation.

SMARTS09

This model demonstrates how to create an accumulating conveyor. Its primary focus is the **Conveyor** module and the **Segment** module.

Modules used: Arrive, Depart, conveyor, Segment, Simulate.

SMARTS10

This model demonstrates how to create an nonaccumulating conveyor. Its primary focus is the **Conveyor** module and the **Segment** module.

Modules used: Arrive, Depart, conveyor, Segment, Simulate.

SMARTS11

In this example, and entity is processed while still remaining on its conveyor. It is centered around the **Conveyor**, **Segment**, and **Advanced Server** modules.

Modules used: Arrive, Advanced Server, Depart, Conveyor, Segment, Simulate.

SMARTS12

In this model, entities that pass an inspection leave the system on one conveyor, and entities that fail leave on a different conveyor. The model focuses on the **Conveyor**, **Segment**, and **Inspect** modules.

Modules used: Arrive, Inspect, Depart, Conveyor, Segment, Simulate, Variable animation.

SMARTS13

This example shows how to model entities of various sizes on an accumulating conveyor. In the model, a large package and a small package stack up on a conveyor while waiting for a bar-code reader. The model utilizes the **Conveyor**, **Segment**, **Server**, and **Chance** modules.

Modules used: Arrive, Server, Chance, Leave, Depart, Conveyor, Segment, Sets, Simulate, Plot animation, Variable animation.

SMARTS14

This example demonstrates how to model a continuous process using constructs from the Blocks and Elements templates. Most important in the model are the **Rates**, **Levels**, and **Detect** blocks and the **Continuous** element. Also used is the **Levels** animation feature.

Modules used: Rates, Levels, Continuous, Detect, Assign, Enter, Leave, Arrive, Depart, Simulate, Level animation.

SMARTS15

In this example, entities have an inventory cost and a holding cost associated with them. These values are displayed on the screen. The modules used in the model are **Assign**, **Variables**, and **Sets**.

Modules used: Arrive, Enter, Process, Assign, Leave, Depart, Sets, Variables, Simulate, Variable animation.

SMARTS16
This model shows how to set up and display a variable that counts the number of parts processed in the system. It utilizes the **Count section of the Depart** module.
Modules used: Arrive, Server, Depart, Simulate, Variable animation.

SMARTS17
This example takes you through the steps of importing a DXF file. It uses the **DXF Import** selection of the File menu.
Modules used: none.

SMARTS18
In this model, one machine experiences a failure based on time, and another fails based on the number of entities it has processed. The focus of the model is the **Failures section of the Server** module.
Modules used: Arrive, Server, Depart, Simulate.

SMARTS19
In this model fork trucks, modeled as guided transporters, carry entities to one of four loading docks. The purpose of the model is to demonstrate how to prevent deadlocking by using Spurs. Its focus is the guided **Transporter** and the **Network Link** modules.
Modules used: Arrive, Enter, Dispose, Transporter, Network Link, Sets, Simulate.

SMARTS20
In this model, a process is broken down into three separate subprocesses. The purpose is to show how modules can be used in a hierarchical way. The central point of the example is the **Access External Logic section of the Process** module.
Modules used: Arrive, Process, Depart, Simulate.

SMARTS21
An entity in this model is not permitted to proceed to the next station unless that station's queue is empty. This is done by sending a signal when that condition is met to the entity waiting upstream. The modules feature in this model are the **Wait**, **Signal**, **Batch**, and **Split**, and the **Global Variable** animation.
Modules used: Arrive, Wait, Batch, Leave, Enter, Split, Server, Choose, Signal, Assign, Delay, Depart, Simulate, Variables, Plot animation, Global Variable animation.

SMARTS22
This model demonstrates how you can route an entity to one of five stations when it exits a module. It is focused on the **Enter**, **Leave**, and **Chance** modules.
Modules used: Arrive, Enter, Process, Chance, Leave, Depart, Simulate.

SMARTS23
In this example, entities are duplicated to represent customers and their orders. The customer and his order are processed separately and then are matched up and sent out of the system. The example features the **Duplicate**, **Match**, and **Batch** modules.
Modules used: Arrive, Duplicate, Delay, Match, Batch, Depart, Sets, Simulate.

SMARTS24
In this example, an entity (customer) is not allowed to proceed to a resource (cashier) unless there are less than three entities already there. The purpose is to show how a "pull" system can be modeled. Featured in the model is the **Overlap Resource portion of the Option section of the Server** module.
Modules used: Arrive, Server, Depart, Resource, Simulate.

SMARTS25

This model demonstrates how to group entities together and send them out of the system. Featured is the **Permanent Batch option in the Batch** module.
Modules used: Arrive, Enter, Batch, Leave, Depart, Simulate.

SMARTS26
This is a model showing shuttle bus utilization at an amusement park. The purpose of the example is to demonstrate how to use more complicated routines, particularly relating to transporters. The model uses the **Search**, **Dropoff**, and **Pickup** blocks, and the **Request** and **Transport** modules.
Modules used: Arrive, Queue, QUEUE, Delay, Assign, Request, Transport, Station, SEARCH, Choose, DROPOFF, DISPOSE, Transporter, Statistics, Variables, Sequences, Simulate, Plots animation, and Variables animation.

SMARTS27
This model demonstrates how to have an entity choose which queue to go to: it selects the shorter queue preceding two Process modules. Its focus is the **PickQueue** module.
Modules used: Arrive, Enter, PickQueue, Leave, Process, Depart, Simulate.

SMARTS28
This model demonstrates how to have an entity choose which station to go to: it selects the station with the fewest entities. The focus of the model is the **PickStation** module.
Modules used: Arrive, Enter, PickStation, Leave, Server, Depart, Simulate.

SMARTS29
In this example, high priority jobs are processed before low priority jobs. The model demonstrates the use of the **Additional Server Information portion of the Options dialog in the Server** module.
Modules used: Arrive, Server, Depart, Simulate.

SMARTS30
This example demonstrates how to read data into a model and how to utilize this information. Its focus is the **Read** module.
Modules used: Arrive, Read, Choose, Server, Depart, Simulate.

SMARTS31
In this example, entities are removed from a queue in no particular order. By doing this, each entity is equally likely to be processed next. The model is centered around the **Remove** block.
Modules used: Arrive, Server, Depart, REMOVE, Simulate, Variable animation.

SMARTS32
This model demonstrates how to set up and use a resource set. The modules featured are **Resource**, **Sets**, and **Advanced Server**.
Modules used: Arrive, Advanced Server, Depart, Resource, Sets, Simulate.

SMARTS33
This example demonstrates how to view entities moving from one station to the next. It shows the use of the **Route option in the Server** module.
Modules used: Arrive, Server, Depart, Simulate.

SMARTS34
Entities in this example have an interarrival rate determined by the number of entities in a queue. This model utilizes the **Scan** block and the **Variables** element.
Modules used: Arrive, Server, Dispose, Create, Scan, Assign, Variables, Simulate, Variable animation, Levels animation.

SMARTS35

This model demonstrates how to use schedules to control the capacity of your resources. It utilizes the **Schedules portion of the Server** module.
Modules used: Arrive, Server, Depart, Simulate, Clock animation.

SMARTS36
This model demonstrates how, if it is late in the day, high priority jobs can be picked out of a queue and sent to a designated worker. The model uses the **Search** and **Remove** blocks, and the **Expressions** element.
Modules used: Arrive, Server, Depart, SEARCH, Choose, Delay, Assign, REMOVE, DISPOSE, Sets, Expressions, Simulate.

SMARTS37
In this model, good boys walk on the sidewalk on their way home and bad boys walk through the grass. The purpose of the model is to demonstrate the use of the **Sequences** module and **Statics** function.
Modules used: Arrive, Server, Depart, Sequences, Sets, Simulate.

SMARTS38
This model demonstrates how to group resources into a set and have entities process on one of the members of that set. The critical modules used are **Sets**, **Resources**, and the **Advanced Server**.
Modules used: Arrive, Advanced Server, Resource, Depart, Sets, Simulate.

SMARTS39
The purpose of this example is to demonstrate how to integrate background objects with animation objects. It uses items from the **Draw** toolbar.
Modules used: Arrive, Enter, Batch, Depart, Transporter, Distance, Simulate.

SMARTS40
This example demonstrates how to pick an entity from a queue and that has been there for a certain amount of time and dispose it. It features the **Search** and **Remove** blocks.
Modules used: Arrive, Server, Depart, Create, SEARCH, Choose, REMOVE, Assign, Dispose, Simulate.

SMARTS41
This model shows how a server can be broken down into subprocesses and animated to show which process is currently being done. It uses the **External Logic feature of the Options section of the Server** module, and the **StateSets feature of the Resource section of the Server** module.
Modules used: Arrive, Assign, Delay, Server, Depart, Variables, Simulate.

SMARTS42
This example demonstrates how to keep track of entities' flowtime in a model. It uses the **Arrive** module and the **Tally portion of the Depart** module.
Modules used: Arrive, Server, Depart, Simulate.

SMARTS43
In this model, entities are batched, sent to a packager, split up, and packaged individually. It utilizes the **Batch** and **Split** modules.
Modules used: Arrive, Enter, Batch, Leave, Split, Process, Depart, Simulate.

SMARTS44
This example shows how to set up a basic free-path transporter. It uses the **Transporter** and **Distance** modules.
Modules used: Arrive, Depart, Distance, Transporter, Simulate.

SMARTS45

In this example, two free-path transporters carry entities from one area to another. It focuses on the **Distance** and **Transporter** modules.
Modules used: Arrive, Depart, Sets, Distance, Transporter, Simulate, Variable animation.

SMARTS46
This model demonstrates how to have an entity process while remaining on a transporter. It uses the **Transporter**, **Distance**, and **Transport** modules.
Modules used: Arrive, Station, Choose, Delay, Assign, Transport, Depart, Sets, Transporter, Simulate.

SMARTS47
In this example, an entity moves to a station on one transporter where it is transferred to another transporter out of the system. The purpose is to show how transporters can interact with each other. The model uses the **Request**, **Free**, **Transport**, **Distance**, and **Transporter** modules.
Modules used: Arrive, Depart, Station, Request, Delay, Free, Transport, Transporter, Distance, Simulate.

SMARTS48
This example demonstrates how different entities can request the same transporter. It is centered on the **Distance** and **Transporter** modules.
Modules used: Arrive, Enter, Batch, Assign, Leave, Queue, Depart, Distance, Transporter, Simulate.

SMARTS49
In this model, the rate entities come into the system depends on what day of the week it is. The modules used are **Arrive** and **Expressions**.
Modules used: Arrive, Server, Depart, Expressions, Simulate, Variable animation.

SMARTS50
Entities in this model enter the system and wait for a signal before they are permitted to proceed. The purpose is to show how entities can be "controlled" by other entities. The model is focused on the **Wait** and **Signal** modules.
Modules used: Arrive, Wait, Server, Create, Signal, Assign, Dispose, Simulate.

SMARTS51
In this model, an entity is created when the user presses a key. This entity alters the capacity of the resource by one unit. The purpose is to demonstrate how the user can interact with a simulation. It uses the **Assign** block and the **Arrivals** element.
Modules used: Arrive, Server, Depart, ASSIGN, DISPOSE, ARRIVALS, Simulate, Variables animation.

SMARTS52
This is an example showing two alternate ways to lay out a plant. The result of each layout is compared to determine which is more efficient. The model utilizes the **Advanced Server**, **Variables**, and **Resource** modules, and **Plots** animation.
Modules used: Arrive, Advanced Server, Depart, Network Link, Transporter, Variables, Sets, Resource, Simulate, Plots animation, Variables animation.

SMARTS53
This model demonstrates how a variable can be used to change how often entities come into a system. This is useful if you have weekly or even seasonal data. The modules used are **Arrive** and **Assign**.
Modules used: Arrive, Assign, Delay, Server, Depart, Variables, Simulate, Variable animation.

SMARTS54
This example demonstrates how a variable can control the amount of time a server is active. It uses the **Create** and the **Schedule portion of the Server** module.
Modules used: Create, Dispose, Arrive, Server, Depart, Variables, Simulate, Variable animation.

SMARTS55

This example demonstrates how to associate a process time with a particular resource instead of with a particular work area. It utilizes the **Recipes**, **Variables**, and **Sets** modules.
Modules used: Arrive, Advanced Server, Depart, Recipes, Variables, Sets, Simulate.

SMARTS56

In this model, resource states are associated with the states busy, idle, and inactive. The amount of time a resource spends in each state is then animated. The model uses the **StateSets** and **Failures options of the Options section in the Server** module, and the **Statistics** module.
Modules used: Create, Server, Dispose, Statistics, Simulate.

SMARTS57

This example illustrates the concept of overlapping. Only when there is space in the next component (conveyor or resource) is an entity permitted to leave the preceding one. The model uses the **External Logic portion of the Options section in the Server** module, and the **Access**, **Exit**, and **Convey** modules.
Modules used: Arrive, Enter, Station, Access, Exit, Convey, Server, Exit, Depart, Segment, Conveyor, Segment, Simulate.

SMARTS58

This example demonstrates how to save statistics to a file. It uses the **Statistics** module and **the Tally portion of the Depart** module.
Modules used: Arrive, Depart, Statistics, Simulate.

SMARTS59

This example shows how to model zero-length guided transporters. They are useful when modeling a large network where collisions are unimportant. It utilizes the **Initial Position Status of the Transporter** module, and **Parking Areas** animation.
Modules used: Arrive, Advanced Server, Depart, Network Link, Transporter, Simulate.

SMARTS60

In this example, an entity is permitted to move to the next resource only if there is room in the queue preceding it. This allows you to create staging areas with limited capacity that block your entities. The model uses the **Proceed**, **Queue**, **Seize**, **Delay**, and **Release** blocks and the **Blockages** element.
Modules used: Arrive, Server, PROCEED, RELEASE, Enter, QUEUE, SEIZE, DELAY, Leave, Depart, BLOCKAGES, Queue, Resource, Simulate.

SMARTS61

This model introduces failures for a conveyor. When the conveyor fails, entities currently moving on it stop, and entities attempting to access it queue up. The model uses the **Stop** and **Start** modules, and **Global Variable** animation.
Modules used: Arrive, Inspect, Depart, Segment, Conveyor, Queue, Create, Stop, Delay, Start, Assign, Dispose, Simulate, Global Variable animation.

SMARTS62

In this model, parts are created via two unique entry points, and are conveyed to a single sorting station. There they first access a single conveyor, then determine which entry point they came from in order to exit the appropriate conveyor. The **Access**, **Exit**, and **Convey** modules are used to accomplish this.
Modules used: Arrive, Enter, Access, Choose, Exit, Convey, Depart, Segment, Conveyor, Simulate.

SMARTS63

This example demonstrates how to count and tally statistics within a Server module. This is done by utilizing the **Count**, **Tally**, and **External Logic portion of the Server** module. Also used is **Variable** animation and **Plots** animation.
Modules used: Arrive, Server, Count, Tally, Depart, Sets, Simulate, Variable animation, Plots animation.

SMARTS64
This example uses the concept of Storages. In the model, an entity is displayed in a single animated storage while it moves through several modules. Used here are the **Store**, **Unstore**, **Storages**, **Wait**, and **Signal** modules, and the **Storages** animation.
Modules used: Create, Signal, Leave, Enter, Store, Delay, Wait, Unstore, Leave, Depart, Storage, Sets, Simulate, Storage animation.

SMARTS65
This module demonstrates the seizing of multiple resource simultaneously. A machine and operator are first seized, then a single delay is incurred, the operator is released and the entity incurs an additional delay before releasing the machine. The **Seize**, **Release**, **Delay**, **Enter**, and **Leave** modules are used in this example.
Modules used: Arrive, Enter, Seize, Delay, Release, Leave, Depart, Resource, Simulate.

SMARTS66
In this model, a transporter fails and is wheeled to a mechanic who fixes it and sends it on. The model requires special logic to allow the failed transporter to seize a resource. The model uses the **Allocate**, **Move**, **Halt**, and **Activate** modules.
Modules used: Arrive, Create, Delay, Allocate, Halt, Move, Free, Process, Activate, Depart, Transporter, Simulate.

SMARTS67
Entities in this model enter a station and alternately choose one of two conveyors. The model focuses on the **Conveyor** module and the **Choose** module.
Modules used: Arrive, Enter, Delay, Choose, Assign, Leave, Depart, Segment, Conveyor, Simulate.

SMARTS68
Process times in this model are set up in a matrix. The proper cell is then referenced in the Server module. The example utilizes the **Expressions** module.
Modules used: Arrive, Server, Depart, Expressions, Simulate.

SMARTS69
In this model, two types of entities enter the system. When an entity enters the server, it checks to see if the previous entity was of a different type. If it was, that entity incurs a setup time. Used here are the **Assign** module, the **External Logic section of the Server** module, and the **Global Variable** animation.
Modules used: Arrive, Server, Assign, Choose, Delay, Depart, Simulate, Global Variable animation.

SMARTS70
An entity in this model will proceed to a machine only if it can begin processing within the next 20 minutes. The example uses the **Choose**, **Assign**, and **Expressions** modules.
Modules used: Arrive, Choose, Leave, Server, Assign, Depart, Expressions, Simulate, Variables animation.

SMARTS71
In this model, entities enter the system at one of four stations. They then move down a conveyor, stopping at all remaining stations, before leaving the system. The model utilizes the **Choose**, **Access**, and **Conveyor** modules.
Modules used: Arrive, Enter, Choose, Advanced Server, Assign, Process, Convey, Depart, Segment, Conveyor, Simulate.

SMARTS72

Entities in this model all move to the next station in unison. This is done by using a non-accumulating conveyor and having the entities process while remaining on the conveyor. The purpose of the example is to demonstrate how to model synchronous movement. It uses the **Advanced Server** and **Conveyor** modules.
Modules used: Arrive, Advanced Server, Depart, Segment, Conveyor, Simulate.

SMARTS73
In this example, entities from one area merge from their conveyor onto another and continue through the system. The example utilizes the **Conveyor** module.
Modules used: Arrive, Enter, Choose, Exit, Access, Leave, Depart, Segment, Conveyor, Simulate.

SMARTS74
Entities in this example choose one of four resources to process on before leaving the system. The purpose of the model is to demonstrate how to define and animate a resource set. It uses the **Pick Station**, **Advanced Server**, **Resource**, and **Sets** modules.
Modules used: Arrive, Pickstation, Route, Advanced Server, Resource, Depart, Sets, Simulate.

SMARTS75
This example model runs for three replications. Entities' process time on the Server module is dependent on which replication the simulation is running. The model utilizes the **Variables** and the **Simulate** modules.
Modules used: Arrive, Server, Depart, Variables, Simulate.

SMARTS76
In this model, a resource is kept busy while an entity transfers; only after the part has completed its transfer is the resource released. The model uses the **Duplicate** and the **Scan** modules.
Modules used: Arrive, Enter, Seize, Delay, Duplicate, Leave, Scan, Release, Dispose, Variables, Resource, Simulate.

SMARTS77
Six different entities enter this model and enter a queue in an order determined by a Critical Ratio. Every 60 time units, this Critical Ratio is reevaluated and the entities are queued up again in a new order. This demonstrates how dynamic reranking of queues can be modeled. The example uses the **QUEUE** block and the **REMOVE** block.
Modules used: Arrive, Assign, QUEUE, SEIZE, Delay, Release, Depart, Choose, REMOVE, QUEUES, RESOURCES, Simulate.

SMARTS78
In this example, the processing time for an entity decreases after every unit is processed. The purpose is to demonstrate how a "learning curve" can be modeled. The model utilizes the **Expressions** module.
Modules used: Arrive, Server, Assign, Depart, Expressions, Simulate.

SMARTS79
Data is normally read into a simulation columns at a time. This example demonstrates how rows of data can be read into a simulation. The model uses the **Read** and **Variables** modules.
Modules used: Create, Read, Assign, Delay, Choose, Dispose, Variables, Simulate.

SMARTS80
Entities enter this system and are assigned one of six sequences to follow, and are assigned a picture based on their sequence. Counters and tallies are kept to track each of the six sequences. The model utilizes the **Sequences** and **Sets** modules.
Modules used: Arrive, Server, Depart, Sequences, Sets, Simulate.

SMARTS81

This model demonstrates how to model a nonstationary poisson process. This method is used to obtain exact results when modeling changing arrival rates. The model uses the **Arrive** and the **Chance** modules.
Modules used: Arrive, Chance, Assign, Count, Depart, Variables, Simulate.

SMARTS82
Three types of entities enter this model and proceed to an Advanced Server. One type enters the first queue, another enters the second queue, and a third enters the third queue. The model uses the **Advanced Server** module, the **Queue** module, and the **Sets** module.
Modules used: Arrive, Advanced Server, Depart, Queue, Sets, Simulate.

SMARTS83
Two types of entities enter this system, and are processed separately. They then enter queues and the two types are matched up, batched, and continue through the system. The model utilizes the **MATCH** block and the **Batch** module.
Modules used: Arrive, Server, Enter, QUEUE, MATCH, Batch, Leave, Depart, QUEUES, Expressions, Simulate.

SMARTS84
This example demonstrates how to read values in from a spreadsheet and use these values in a simulation. It uses the **Read** module.
Modules used: Create, Read, Dispose, Arrive, Server, Depart, Simulate.

SMARTS85
This example shows one way to model a limited-capacity buffer in front of a resource. It utilizes the **Server** module and the **Resource** module.
Modules used: Arrive, Server, Depart, Resource, Simulate.

SMARTS86
This example demonstrates an accounting process where invoices arrive and are sent to a processing clerk. Within the **Actions** module, the entity seizes a resource, delays for a processing time, assigns a variable, and releases the resource.
Modules used: Arrive, Actions, Leave, Resource, Statistics, Animate, Simulate.

SMARTS87
This example shows the use of the **Actions** module for material handling applications. The process demonstrated here differs from the Server, Advanced Server, and Inspect modules because it allows the modeler to determine the order of the actions taken.
Modules used: Arrive, Actions, Storage, Conveyor, Segment, Transporter, Distance, Simulate.

SMARTS88
In this model, the entities exit a conveyor, seize a resource, delay for processing, release the resource, send a signal, are unbatched, and a variable is assigned all through the **Actions** module.
Modules used: Arrive, Choose, Assign, Wait, Leave, Actions, Expressions, Variables, Sets, Simulate.

SMARTS89
This example shows the use of single-level menus. A menu appears at the beginning of each simulation run prompting the user to enter information about processing, arrival, and travel times. These values are then used in the Arrive and Server modules. The purpose is to show how the **Menu** module can be used.
Modules used: Arrive, Menu, Server, Depart, Variables, Simulate.

SMARTS90

This model shows the use of multiple level menus. The main menu prompts the user for travel time information and sends the user to one of two second level menus. One prompts for arrival information, while other prompts for processing time. To accomplish this, three **Menu** modules are used.
Modules used: Arrive, Menu, Server, Depart, Variables, Sets, Simulate.

SMARTS91
When entities arrive in this example, they check to see if the queue for the main server is greater than 2. If it is, a menu will appear that allows the user to interactively specify whether to send the entity to the main or backup processing area. The **Menu** module is the focus of the example.
Modules used: Arrive, Menu, Server, Depart, Choose, Leave, Simulate.

SMARTS92
This example demonstrates how to animate information such as number in queue, average utilization of a resource, system time, and entity flowtime. The model uses the **Animate** module.
Modules used: Arrive, Server, Depart, Animate, Simulate.

SMARTS93
This model demonstrates how to simulate a continuous process. In the model, there is a "Source" container, a "Transfer" container, and a "Sink" container. Liquid begins in the Source, flows through the Transfer, and ends up in the Sink. The model uses the **Container** module.
Modules used: Container, Leave, Depart, Simulate.

SMARTS94
In this model, paint flows from two unique source containers into a single Transfer container where it is mixed. It then continues to a Sink container. The model utilizes the **Container** module.
Modules used: Container, Statistics, Simulate.

SMARTS95
In this model, an operator is used to fill an entry container when it is empty. When the Mixer container is full, a count and tally are done to keep statistics on the full containers. The model focuses on the **Container** module.
Modules used: Container, Arrive, Process, Count, Tally, Simulate

SMARTS96
This model demonstrates how to have a replication run for a specific amount of time and there are no entities left in the system. This is usually the case if you the model represents any service industry, such as a bank or restaurant. The key to this model is the use of the **ARRIVALS** element.
Modules used: Enter, Leave, Server, Inspect, Depart, Arrivals, Simulate

SMARTS97
This model demonstrates how to have a replication run for a specific amount of time and there are no entities left in the system. This is usually the case if the model represents any service industry, such as a bank or restaurant. The key to this model is the use of the Count Limit in the **Statistics** module.
Modules used: Arrive, Choose, Leave, Enter, Count, Dispose, Variables, Inspect, Server, Statistics, Simulate

SMARTS98
In this example, an end-user is prompted to enter the parameters for the distributions used for interarrival rate and process time. These distributions can also be altered at any point during the run. The model utilizes Arena's **Menu** module.
Modules used: Arrive, Server, Depart, Menu, Variables, Simulate

SMARTS99

It is not uncommon for a system to have an unlimited supply of entities; every time an entity is processed at the initial station, another entity is available to enter the system. This example demonstrates how to model this scenario. The trick to this model is the use of the **Duplicate** module.
Modules used: Create, Assign, Route, Server, Duplicate, Depart, Simulate

SMARTS100
This example runs until 1000 parts have been produced. The purpose is to show how replications can be ended based on conditions other than time. It uses the Count Limit in the **Statistics** module.
Modules used: Arrive, Server, Depart, Statistics, Simulate

SMARTS101
Entities in this example can be processed by either of the two resources. However, one type of entity prefers one resource, and the other type prefers the other resource. The key to this model is the use of resource sets defined in the **Sets** module
Modules used: Arrive, Seize, Delay, Assign, Release, Route, Depart, Resource, Statistics, Sets, Simulate

SMARTS102
In this example, the first part is sent to Station 1, the second to Station 2, the third to Station 3, the fourth to Station 4, the fifth to Station 1, etc. The purpose is to demonstrate how entities can be routed to their destination cyclically. The model utilizes Arena's **Route** module.
Modules used: Arrive, Assign, Choose, Route, Server, Depart, Variables, Sets, Simulate

SMARTS103
The entities in this example are conveyed to a processing station, where they remain on the conveyor but their size (thus their space on the conveyor) is reduced. They then continue through the system. This model uses the **EXIT** module from Arena's Blocks panel.
Modules used: Arrive, Enter, EXIT, Assign, Convey, Depart, Animate, Conveyor, Simulate

SMARTS104
Entities requesting transporters get the transporter first based on priority. Then, the tie-breaker is the shortest distance. This example demonstrates how to allow the entity waiting longest to get the transporter. This model uses the **Request** module (or it's equivalent inside the Arrive module).
Modules used: Arrive, Depart, Distance, Transporter, Simulate

SMARTS105
Arena shows you how often each individual resource fails by default. If you want to know how often ANY of a number of resources fails, you can also retrieve that information. This is useful if, for example, you need to know how often a line goes down. The model features the **Expressions** module.
Modules used: Arrive, Server, Depart, Expressions, Statistics, Simulate

SMARTS106
In this example, we don't want both of our tallies to be written to the standard summary report, we want only one to be written. However, there is no way to remove specific entries directly from the .out file. This example shows how we can, in effect, do the same thing. The **REPORTS** module is used.
Modules used: Create, Duplicate, Delay, Tally, Dispose, TALLIES, REPORTS, PROJECT, REPLICATE

SMARTS107
In many systems, a certain percentage of each part is discarded as scrap. This example demonstrates a way to model this scrap. The model uses the **Batch** and **Split** modules.
Modules used: Arrive, Batch, Leave, Server, Split, Choose, Count, Dispose, Sets, Simulate

SMARTS108

This model demonstrates how to "shut off" entity creations once a specific condition is met. This model uses the Max Batches field in the **Create** module.
Modules used: Create, Count, Dispose, Simulate

SMARTS109
By default, Arena calculates resource utilization for the entire replication length. In this model, resource utilization is calculated written to the output report, and reinitialized every hour. Of primary importance are the **Statistics** and the **Write** modules.
Modules used: Arrive, Server, Assign, Depart, Create, Write, Dispose, Statistics, Variables, Simulate

SMARTS110
In this example there are three types of entities. When any entity enters the system, it waits until its type is requested before it is permitted to continue. The model uses the **Wait** and **Signal** modules.
Modules used: Arrive, Enter, Wait, Leave, Depart, Create, Assign, Signal, Dispose, Simulate

SMARTS111
In this model, first a type 1 entity enters the system, then a type 2 enters, then a type 1, then a type 2, etc. The purpose of the model is to demonstrate how alternating entity types can be easily modeled in Arena. The model utilizes the **Arrive** module and the **Choose** module.
Modules used: Arrive, Choose, Assign, Leave, Depart, Simulate

SMARTS112
This model allows the user to change the simulation run length during the run. Every time the specified ending time is reached, the user receives a menu asking if the run should be extended. This model uses Arena's **Menu** module.
Modules used: Create, Delay, Duplicate, Menu, Arrive, Server, Depart, Simulate

SMARTS113
In this example, a worker is seized at one of two stations. The worker correctly animates at the station he is located. This purpose of the model is to demonstrate the use of positional resources in Arena. The model features the **Resources dialog in the Server** module.
Modules used: Arrive, Server, Depart, Simulate

SMARTS114
When the half-width around the Time in System tally in this model is less than or equal to a certain value, the replication will immediately end. This feature is useful if you want to stop a replication when a certain statistic is "good enough" to begin output analysis. The model it centered around the **Simulate** module.
Modules used: Arrive, Server, Depart, Simulate

SMARTS115
This Visual Basic utility program allows you to create an array of animation status variables by supplying the variable name and the maximum array index. For example, specifying Inventory for the variable name and 10 for the maximum array index would cause 10 animated variables, Inventory(1) to Inventory(10) to be placed in the model. The purpose of the model is to show the use of Arena's **OLE Automation** feature by creating an external program (in this case, a Visual Basic program) that automates Arena.
Modules used: Variable animation

SMARTS116
Arena's ability to use OLE technology (Object Linking and Embedding) allows various types of "objects" to be inserted (embedded) into your simulation models. There are many types of objects that can be used, and ClipArt is just one example. This Smart File intends to demonstrate ways that **ClipArt** can be used in a model.
Modules used: Arrive, Server, Depart, Simulate

SMARTS117

This model depicts three workstations with unique resources and processing times. Sequences are used to ensure that each entity type is routed according to its processing sequence. Indexed variables are used to store the correct process time for each entity type at each workstation. Sets are used to model all three workstations in a single AdvServer module. The purpose of the model is to show you how to use the **Module Data Transfer wizard** to import the variable values from an **Excel spreadsheet** into the Variables module.
Modules used: Arrive, Chance, Assign, Route, Advanced Server, Depart, Sequences, Sets, Simulate, Variables

SMARTS118
This model shows an order processing system where the process varies according to the order type. The purpose of the model is to show you how to use the **Module Data Transfer wizard** to import the variable values from a **Text file** into the Sequences module.
Modules used: Arrive, Server, Depart, Sequences, Simulate

SMARTS119
This model depicts a basic processing station where the processing resource can be IDLE, BUSY, or JAMMED. Ten (10) replications are performed, keeping output statistics on the number of entities processed in each replication. The purpose of the model is to show the use of the **Visual Basic for Applications** interface to save the summary report in a standard file format called CSV (Comma-Separated Value.)
Modules used: Create, Leave, Enter, Process, Count, Tally, Dispose, Statistics, Simulate

SMARTS120
This model demonstrates how to automate Excel from Arena. When the model has run to completion, Arena opens Excel and creates a chart for the amount of time the resource is busy, idle, and jammed. The purpose of the model is to demonstrate another use of **Visual Basic for Applications** from within Arena.
Modules used: Arrive, Enter, Process, Leave, Depart, Statistics, Simulate

SMARTS121
This model depicts a basic order processing system where orders are first consolidated into an order batch size. The purpose of the model is to show the use of the **Visual Basic for Applications** interface to display a form allowing users to define multiple scenarios for the simulation run.
Modules used: Create, Leave, Enter, Batch, Process, Count, Tally, Dispose, Sets, Animate, Variables, Simulate

SMARTS122
This model depicts an Inspection station where entities that pass inspection exit the system, while entities that fail inspection are reworked and reinspected. When you click Go, a dialog appears which allows you to change the Inspection time and failure probability. This information is then stored in the modules, the model is checked, and the simulation begins. The purpose of the model is to show the use of the **Visual Basic for Applications** interface to display a form allowing users to replace data in the modules.
Modules used: Arrive, Inspect, Depart, Sets, Server, Assign, Leave, Simulate

SMARTS123
This model depicts two conveyors merging into one, with the stipulation that only one entity may occupy the merge point at any given time. Packages enter via two entry areas and convey towards the junction. The junction (modeled as a resource) is seized, and entities are conveyed to the Sorting station. The entity is then conveyed just past the junction, where it releases the junction and conveys to the Done station. The purpose of the model is to show how to display a model description using the **Visual Basic for Applications** interface.
Modules used: Arrive, Enter, Seize, Convey, Access, Exit, Choose, Release, Depart, Simulate, Segment, Conveyor

SMARTS124

This model plays a sound whenever the number of customers waiting in the Worker 1_Q is greater than three. When an entity enters the VBA module, the code in the VBA_Block_1_Fire event is executed, playing the sound. After opening the model, click on the Visual Basic Editor button to view the code. The purpose of the model is to show the use of the **VBA** block to execute **Visual Basic for Applications** code.
Modules used: Create, Leave, Enter, Process, Count, Tally, Dispose, Scan, VBA, Simulate

SMARTS125
This model depicts a simple process where entity arrivals and process times are read from an Excel spreadsheet. After processing, parts are shipped and their part number and ship time is written to an Excel spreadsheet. The purpose of this model is to show you how to read and write directly to an Excel spreadsheet using the VBA module.
Modules used: Arrive, Assign, Delay, VBA, Duplicate, Server, Variables, Enter, Count, Dispose

SMARTS126
This Visual Basic Utility program creates a quick Arena model using Arrive, Server, Depart and Simulate modules. It also places animated routes between the modules, saves the model, and optionally runs the simulation. The user is prompted for all necessary information. The purpose of the model is to show the use of Arena's **OLE Automation** feature by creating an external program (in this case, a Visual Basic program) that automates Arena.
Modules used: Arrive, Server, Depart, Simulate

SMARTS127
Sometimes when you are modeling batches of entities, it is necessary to manipulate the processing area in one of two ways:
1. Create a control so that the number of batches permitted in the processing area at a time is restricted.
2. Allow the individual members of the batch to be processed individually, even though they are to flow as a batch.
This model demonstrates both ways to process a batch of entities. It is focused on the **Batch** and **Split** modules.
Modules used: Arrive, Batch, Leave, Station, Seize, Split, Delay, Release, Dispose, Simulate